AD-A115 840 UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFO--ETC F/G 9/2
STATE DELTAS THAT REMEMBER: A FORMALISM FOR DESCRIBING STATE CH--ETC(U)
MAY 82 L MARCUS
UNCLASSIFIED ISI/RR-81-93 DAHC15-72-C-0308 NL

END
DATE FILMED
7-82
DTIC

ISI/RR-81-93

May 1982

Leo Marcus

State Deltas that Remember: A Formalism for Describing State Changes

INFORMATION SCIENCES INSTITUTE

UNIVERSITY OF SOUTHERN CALIFORNIA

4676 Admiralty Way/Marina del Rey/California 90291

(213) 822-1511

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ISI/RR-81-93	2. GOVT ACCESSION NO. AD-A115 844	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* State Deltas that Remember: A Formalism for Describing State Changes		5. TYPE OF REPORT & PERIOD COVERED Research Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Leo Marcus		8. CONTRACT OR GRANT NUMBER(s) DAHC 15 72 C 0308 F30602 78 C 0008
9. PERFORMING ORGANIZATION NAME AND ADDRESS USC/Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90291		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE May 1982 13. NUMBER OF PAGES 13
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* ----------		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

This document is approved for public release and sale; distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

----

18. SUPPLEMENTARY NOTES

----

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

program logic, state changes, state deltas, temporal logic

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

We define a system for describing state changes based on the state deltas of S. Crocker. This approach combines the "sometimes" assertion method with a limited "during" modality.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ISI/RR-81-93
May 1982

Leo Marcus

State Deltas that Remember: A Formalism for Describing State Changes

INFORMATION SCIENCES INSTITUTE

UNIVERSITY OF SOUTHERN CALIFORNIA

4676 Admiralty Way/ Marina del Rey/ California 90291

(213) 822-1511

This research was supported in part by the Rome Air Development Center under Contract No. F30602 78 C 0008 and in part by the Defense Advanced Research Projects Agency under Contract No. DAHC15 72 C 0308. Views and conclusions contained in this report are the author's and should not be interpreted as representing the official opinion or policy of RADC, DARPA, the U.S. Government, or any other person or agency connected with them.

CONTENTS

ACKNOWLEDGMENTS

I wish to thank Steve Crocker and the other members of the Microverification project at ISI for many stimulating discussions.

1. INTRODUCTION

We define a system for describing state changes, briefly as follows: Our intention is to talk about state changes in a generalized machine; that is, as a first approximation, state change means a new interpretation of constant symbols in a given relational structure. The language for describing these state changes allows sentences of the following kind, called "state deltas," to be formed: If the system is in a state satisfying a certain "precondition" and certain of the constant symbols (the "environment") have not changed since a given previous time, then there is a later time at which the state will satisfy a certain "postcondition" and during the elapsed time the interpretations of only certain specified constant symbols (in the "modification" list) are allowed to change. This restricted use of a "since-during" modality is one of the special features of this approach.

The precondition and postcondition may themselves be (or contain as conjuncts) sentences like the above, and the postcondition may contain function symbols that are interpreted as referencing any previous values of constants. The ability (necessity here) to remember previous values is the other special feature.

There are two complementary ways to interpret the use of state deltas as subsentences of other state deltas. First, from the viewpoint of an omniscient observer with the whole history and future of state changes laid out before him, the truth of any state delta at any time can be checked. Or from the viewpoint of the control of an executing program, a state delta can be understood as a program, and so its truth at a certain time simply means that that program is available for execution.

Our formulation is directly based on the state deltas introduced by S. Crocker in his thesis [1]. It also bears a close relation to elements of the "sometimes assertion" method [3] with a restricted "during" modal operator, and a more distant relation to the work of Pratt [5] and Pnueli [4] and others in modal and temporal logic as applied to computer science. A discussion of similar ideas as applied to natural language appears in Saarinen [6].

The reasons for focusing on this particular "during" modality, i.e., that during a certain interval certain constants do not change value, are several: first, it is an efficient abbreviation for saying that all the old facts known about the unchanging constants are still true in the new state; it is a useful property in the context of parallel computations to know that certain constants (i.e., program variables) have not changed their values during a certain interval, and thus may be referenced by some other process any time in that interval. This is of course stronger than just knowing that the value at the end of the interval is the same as at the beginning.

A system for checking proofs of microcode correctness based on state deltas has been implemented at USC Information Sciences Institute and is described in [2].

2. DEFINITIONS

First we are given a totally ordered set $A = \langle A, \leq \rangle$ with minimum element START $\leq t$, for all $t \in A$, and no maximal element. This is regarded as time, along which the state of a computation may change.

Next we are given an arbitrary first-order language L, $\leq \notin L$, and an L-model B. B is the model of the "background data domain and architecture." Let Ω be a finite set of constant symbols with $\Omega \cap L = \emptyset$. These constants are the "program identifiers" or machine "place names." Here, the constants are assumed to represent disjoint places. Of course, it is possible to allow places to intersect, but this adds unessential complications to the presentation. Finally we are given the function symbols "." (read "dot"), $\Box$, and $\Box_n$ for $n < \omega$ from Ω to the universe of B. You may think of dot as mapping a place to its (current) "contents" and $\Box_n$ to its "n-th previous contents." We write $\Box$ instead of $\Box_1$ (and . instead of $\Box_0$.) Thus $\Box c$ is the previous value of c. $\Box_2$ is the value before that, etc. Let L^* be $L \cup \{., \Box\} \cup \Omega$, and $L_1 = L^* \cup \{\Box_n : n < \omega\}$.

A word on the utility of dot. Essentially, dot is a way of making explicit the interpretation of the elements of Ω in B. That is, instead of the usual assumption that the program change the

interpretation of the constant symbols, here the constants do not change their interpretation (i.e., each represents one piece of hardware for the whole computation), but rather the function dot changes its values.

We consider partial (machine or program) states, in the sense that the image under dot of some elements of Ω may not be completely specified. However, we may have some information in the form of a set of sentences S of L_1. By assuming that the elements of B are first-order definable in B (or at least that this is true for the elements of B that are possible images of Ω under dot), we can completely specify .c for $c \in \Omega$ by a sentence of L_1. But we do not restrict S to contain only defining sentences of this form.

Of course we want S to be consistent with B. In addition, there may be a set of sentences T of $L_2 \supseteq L_1$ that we want to hold always (in all partial states), and S must also be consistent with T. For example, relations such as the length of a place (considered as a register in a machine) or inclusions among several places are "architectural" facts that should not change during a given computation. Now we are ready for the definition.

Definition: A partial state (for the system with "background" model B and "architecture" sentences T) is a set S of sentences of L_1, closed under logical deduction, such that there exists an expansion B^* of $B \cup \Omega$ satisfying $B^* \models SUT$, where Ω can be considered just as the set Ω with equality. (See remark below for an alternative to demanding closure under logical deduction; this is not an essential requirement: in implementations, S is of course always finite.) Note that inside a given partial state there is no restriction on the relations among . and $\Box_n$ for all n. But see (4) and (5) below.

Definition: First-order satisfaction in partial states. Let S be a partial state and φ be a sentence in L_2. $S \vdash \varphi$ ("φ follows from S") iff for every expansion B^* of $B \cup \Omega$ that satisfies $B^* \models SUT$, also $B^* \models \varphi$.

So, for example, if φ is a sentence of L, then $S \vdash \varphi$ iff $B \models \varphi$.

The next definition generalizes $c \in C_1$.

Definition: Let $C_1 \subseteq \Omega$, $c \in \Omega$. C_1 determines c (with respect to T) if there are $c_1,\ldots,c_n \in C_1$ and a function $f(x_1,\ldots,x_n)$ definable in T such that $T \vdash f(c_1,\ldots,c_n) = .c$. C_1 determines $C_2 \subseteq \Omega$ if for every $c \in C_2$, C_1 determines c.

Thus, if the values (or contents) of C_1 are preserved, then the same is true for C_2.

Definition: A state-delta model is a system A^* consisting of a time model $A = \langle A, \leq \rangle$, a background model B in language L (for every $t \in A$ a partial state S_t in language L_1) a set of sentences T in language L_2 (all as above), and in addition for every interval $I = [t_0,t_1]$ of A, a subset $C_I \subseteq \Omega$ (to be thought of as containing those constants whose contents do not change over I) such that (1) through (5) below hold.

(1a) If $I \subseteq J$, then $C_J \subseteq C_I$ (If I is a one-point interval or if it is empty, then $C_I = \Omega$.) (1b) $I \cap J \neq \emptyset$ implies $C_I \cap C_J \subseteq C_{I \cup J}$. Notice that for all I,J $C_I \cap C_J \supseteq C_{I \cup J}$ already follows from (1a). So actually we have that $I \cap J \neq \emptyset$ implies $C_I \cap C_J = C_{I \cup J}$.

Definition: $c \in C$ changes value (perhaps) at t_1 if there exists $t_c < t_1$ such that for every t, $t_c \leq t < t_1$, c is determined by $C_{[t_c, t]}$, but not by $C_{[t_c, t_1]}$.

(2) For every constant c, the sequence of times at which c changes is of order type $\leq \omega$, and if it is infinite, then it is cofinal in A.

Definition: $c^{-1}(t) = \max(\{t' : t' \leq t,\ c \text{ changes at } t'\} \cup \{\text{START}\})$ and $c^{-(n+1)}(t) = \max(\{t' : t' < c^{-n}(t),\ c \text{ changes at } t'\} \cup \{\text{START}\})$. (Notice $c^{-1}(t) = t$ if c changes at t, but $c^{-(n+1)}(t) < c^{-n}(t)$ unless $c^{-n}(t) = \text{START}$).

$c^1(t) = \min(\{t' : t' > t,\ c \text{ changes at } t'\} \cup \{\omega\})$ and $c^{n+1}(t) = \min(\{t' : t' > c^n(t),\ c \text{ changes at } t'\} \cup \{\omega\})$.

(3) If $\varphi(c_1,\ldots,c_n)\in S_{t_0}$, then $\varphi(c_1,\ldots,c_n)\in S_t$ for all t such that $c_1,\ldots,c_n$ do not change between t and t_0 (or between t_0 and t); that is, $\max\{c_1^{-1}(t_0),\ldots,c_n^{-1}(t_0)\} \leq t < \min\{c_1^{1}(t_0),\ldots,c_n^{1}(t_0)\}$.

Now we come to state the connections between present and previous values. We write down the condition only for $\Box$ and leave the obvious but messy case of $\Box_n$ for the reader.

(4) If $\varphi(.c_1,\ldots,.c_k,\Box c_{k+1},\ldots\Box c_n)\in S_{t_0}$, then $\varphi(.c_1,\ldots,.c_k,.c_{k+1},\ldots,.c_m,\Box c_{m+1},\ldots,\Box c_n)\in S_t$ for all t such that $\max\{c_1^{-1}(t_0),\ldots,c_k^{-1}(t_0),\ c_{m+1}^{-1}(t_0),\ldots,c_n^{-1}(t_0),c_{k+1}^{-2}(t_0),\ldots,c_m^{-2}(t_0)\} \leq t < \min\{c_{k+1}^{-1}(t_0),\ldots,\ c_m^{-1}(t_0)\}$.

(5) If $\varphi(.c_1,\ldots,.c_k,\Box c_{k+1},\ldots,\Box c_n)\in S_{t_0}$, then $\varphi(.c_1,\ldots,.c_j,\Box c_{j+1},\ldots,\Box c_k,\Box c_{k+1},\ldots,\Box c_n)\in S_t$ for all t such that $\max\{c_{j+1}^{1}(t_0),\ldots,c_k^{1}(t_0)\}\leq t<\min\{c_1^{1}(t_0),\ldots,c_j^{1}(t_0),c_{j+1}^{2}(t_0),\ldots,c_k^{2}(t_0),c_{k+1}^{1}(t_0),\ldots,c_n^{1}(t_0)\}$.

1. Statement (1) says that a constant is preserved over a given interval I iff it is preserved over each two nondisjoint subintervals whose union is I.

2. Statement (2) outlaws "Zeno machine" calculations and allows you to count backward to the nth previous change of a constant.

3. Statement (3) says that if the values (or contents) of $c_1,\ldots,c_n$ are (forced to be) preserved during an interval, then every partial state attached to a time in that interval contains the same information about $c_1,\ldots,c_n$.

   Remark If we did not have closure under logical deduction, we would have to write $S_{t_0}\vdash\varphi$ and $S_t\vdash\varphi$ instead of $\varphi\in S_{t_0}$ and $\varphi\in S_t$ in (3), (4) and (5).

4. Statement (4) says that all the information about previous contents is derived from previous information about (then-) present contents.

5. Statement (5) says that when the contents of a constant changes, whatever was known about its "present contents" (.) is now known about its "previous contents" ($\Box$).

Now we define state deltas (P,E⇒Q,M) which will mean the following: if P is true in a certain "environment" E, then Q will be true later, and along the way the values of constants outside M were not modified. Note that ⇒ is used for state changes, and → for logical implication.

We allow . and $\Box$ to appear in state deltas, but not $\Box_n$ for $n>1$. In addition, a first-order sentence Q containing $\Box$ must appear in the postcondition of the state delta immediately containing Q. This

conforms to the view that $\Box$ (previous) always relates to the value at the time of the precondition. This also explains why we do not allow $\Box_n$ for $n>1$. A state delta can know only about one level of "previous." The conditions on $\Box_n$ come into play because of the "nesting" of state deltas. We will see below that if in fact c did not change from the time of the precondition to the time of the postcondition, then $\Box c$ in the postcondition is the same as .c. If this causes a contradiction, then the interpretation is that the computation is aborted. For example, if Q implies that $.c \neq \Box c$. but $c \notin M$, then when $(P,E \Rightarrow Q,M)$ is "applied," the computation aborts.

Definition: SD, the set of state deltas, is the smallest set such that

1. If $E,M \subseteq \Omega$, P,Q, are first order in L^*, P does not have an occurrence of $\Box$, then $(P,E \Rightarrow Q,M) \in SD$.
2. If $E,M \subseteq \Omega$, $P,Q, \in SD$, then $(P,E \Rightarrow Q,M) \in SD$.
3. If $P,Q \in SD$, then $P \wedge Q$, $P \vee Q$, $\neg P \in SD$.

The truth value of a state delta changes as a function of time, or more precisely with respect to S_t. A state delta may be viewed as a formula $(P,E \Rightarrow Q,M)(t)$.

Satisfaction for state deltas is defined in state delta models A^* as defined above.

First we have to tell how to translate an occurrence of $\Box$ into . or the appropriate $\Box_n$.

Definition: Let $t_1 \leq t_2$.

1. If Q is first order in L^*, then $Q_{[t_1,t_2]}$ is the sentence of L_1 obtained from Q by replacing every occurrence of $\Box c$ by $\Box_n c$ where n is the number of times c changed value in $[t_1,t_2]$, or by .c ($= \Box_0 c$).
2. $(P,E \Rightarrow Q,M)_{[t_1,t_2]} = (P,E \Rightarrow Q,M)$. (no change)
3. $(P \wedge Q)_{[t_1,t_2]} = P_{[t_1,t_2]} \wedge Q_{[t_1,t_2]}$, $(P \vee Q)_{[t_1,t_2]} = P_{[t_1,t_2]} \vee Q_{[t_1,t_2]}$, $(\neg P)_{[t_1,t_2]} = \neg P_{[t_1,t_2]}$.

Now we can define

$$A^* \models (P,E \Rightarrow Q,M)(t_0) \text{ if and only if}$$

$$(\forall t_1 \geq t_0)\left[(P(t_1) \wedge E \subseteq C_{[t_0,t_1]} \rightarrow (\exists t_2 \geq t_1)\,(Q_{[t_1,t_2]}(t_2) \wedge \Omega \cdot M \subseteq C_{[t_1,t_2]})\right].$$

(Note: The above definition was written in "logical notation" for convenience. It is not implied that this is really a first-order sentence. If P is a first-order sentence, then P(t) means $S_t \vdash P$.) In words: the state delta is true at time t_0 if for every later t_1 at which P is true, and for which the environment has not changed between t_0 and t_1, there is a still later t_2 at which Q is true, and for which the interpretation of constant symbols outside of the modification list has not changed between t_1 and t_2. Notice that the calculation of $Q_{[t_1,t_2]}$ is postponed until Q is first order. This is so that the time of the precondition (t_0) will already be known. For example, in

$$(P_1(.c), E_1 \Rightarrow Q_1(\Box c) \wedge (P_2(.c), E_2 \Rightarrow Q_2(\Box c), M_2), M_1)$$

the two $\Box c$'s do not refer to the same object.

3. SOME FACTS

The following are some easily verified facts about state deltas:

1. $\models (\forall t_1 \geq t_0)\left[(P,E \Rightarrow Q,M)(t_0) \wedge E \subseteq C_{[t_0,t_1]} \rightarrow (P,E \Rightarrow Q,M)(t_1)\right]$

   That is, if a state delta is true at t_0 and the environment does not change through t_1, then the state delta is true at t_1.

2. $\models E \subseteq E' \wedge M \subseteq M' \wedge \forall t(P'(t) \rightarrow P(t)) \wedge \forall t(Q(t) \rightarrow Q'(t)) \rightarrow \forall t((P,E \Rightarrow Q,M)(t) \rightarrow (P',E' \Rightarrow Q',M')(t))$

   That is, enlarging the environment and modification list, strengthening the precondition, and weakening the postcondition preserve satisfaction.

3. $\models \forall t\left[(P,\Omega \Rightarrow Q,\varnothing)(t) \equiv (P(t) \rightarrow Q(t))\right]$;

   in particular, $\models \forall t\left[(\forall x(x = x),\Omega \Rightarrow P,\varnothing)(t) \equiv P(t)\right]$.

4. $\models \forall t\left[(P,\varnothing \Rightarrow Q,\Omega)(t) \equiv (\forall t_1 \geq t)(P(t_1) \rightarrow (\exists t_2 \geq t_1)Q(t_2))\right]$.

5. $\models \forall t\left[(P,\Omega \Rightarrow Q,\Omega)(t) \equiv (P(t) \rightarrow (\exists t_1 \geq t)Q(t_1))\right]$.

6. $\models \forall t\left[(P,\varnothing \Rightarrow Q,\varnothing)(t) \equiv (\forall t_1 \geq t)(P(t_1) \rightarrow Q(t_1))\right]$

7. $\models \forall t\left[\left((P,E \Rightarrow P_1,M)(t) \wedge (\forall t_1 \geq t)(E \subseteq C_{[t,t_1]} \rightarrow (P_1,\Omega \cdot M \Rightarrow Q,M)(t_1))\right) \rightarrow (P,E \Rightarrow Q,M)(t)\right]$;

in particular:

8. $\models (\forall t)\ [(P,\Omega\Rightarrow Q_1,M)(t) \wedge (Q_1,\ \Omega\text{-}M\Rightarrow Q_2,\ M)(t) \wedge \ldots \wedge (Q_{n-1},\ \Omega\text{-}M\Rightarrow Q_n,\ M)(t) \rightarrow (P,\Omega\Rightarrow Q_n,\ M)(t)]$.

9. $\models (\forall t)[(P_1,E_1\Rightarrow Q_1,\ M_1) \wedge (P_2,\ E_2\Rightarrow Q_2,\ M_2) \rightarrow (P_1 \vee P_2,\ E_1 \cup E_2\Rightarrow Q_1 \vee Q_2,\ M_1 \cup M_2)]$;

in particular,

10. $\models (\forall t)\ [(P\wedge P',\ E\Rightarrow Q,M) \wedge (P\wedge \sim P',\ E\Rightarrow Q,M) \rightarrow (P,E\Rightarrow Q,M)]$.

11. $(P_1,E\Rightarrow(P_2,\Omega\Rightarrow Q,M),\varnothing) \rightarrow (P_1\&P_2,E\Rightarrow Q,M)$

(Instead of $\varnothing$ it is sufficient to have any subset of M disjoint from the places affecting P_2, and instead of Ω any set at all will suffice.)

12. If Q is first order and $c \notin M$, then

$$\models(\forall t)[(P,E\Rightarrow Q,M)(t)\equiv(P,E\Rightarrow Q'.M)(t)]$$

where Q′ is obtained from Q by replacing all occurrences of $\Box c$ by .c (and leaving occurrences of .c as they are).

Thus, for example, a first-order Q, interpreted as a state delta, is $(\forall x(x = x),\Omega\Rightarrow Q,\varnothing)$ by 3, and thus by 12, every occurrence of $\Box$ in Q is interpreted as . (dot).

Now we state a general induction principle which can be used to derive one state delta from another:

13. Let R(x,y) be a well-founded partial order, and $E\cap M = \varnothing$.

$\models (\forall t)\ [(Q\wedge\exists xR(x,.c),E\Rightarrow\ Q\wedge R(.c,\Box c),M)(t) \rightarrow (Q\wedge\exists xR(x,.c),E\Rightarrow\ Q\wedge\neg\exists xR(x,.\ .c),M)(t)]$.

4. TEMPORAL POWER OF STATE DELTAS

First, let us make a short comparison with some of the operators of classical temporal logic. While at first it may seem as though state deltas can only claim that at some time in the future the desired situation will be attained, much more can be stated through proper use of the environment and modification lists.

For example,

"Q will always be true in the future" is

$$(True,\varnothing \Rightarrow Q,\varnothing).$$

Let us examine "P is true until Q." If the set of places P depends on, Ω_P, is disjoint from those of Q, Ω_Q, then we can write

$$P\&(True,\Omega \Rightarrow Q, \Omega_Q).$$

However, it seems impossible if P and Q have places in common, since the only obvious way to make sure P stays true is to make its places unmodifiable. But then that restricts, or prohibits, Q's changing.

The following question also arises: How does one know in (P,E⇒Q,M) when the postcondition time has arrived? One may know there is a time in the future when the output will be ready, but how does one know when to look?

We can solve this problem by adding an auxiliary place SIGNAL to the language. The following nested state delta guarantees that if you look at the state when SIGNAL is ON (assuming it is OFF at the time of the precondition), Q will be true:

$$(P,E\Rightarrow Q\&(True,\emptyset\Rightarrow SIGNAL = ON,\{SIGNAL\}),M).$$

Thus, Q becomes true sometime, and then with Q held constant SIGNAL becomes ON. Notice that SIGNAL cannot become ON between P and Q since it is not a member of M.

REFERENCES

[1] Crocker, S., *State Deltas: A Formalism for Representing Segments of Computation*, Ph.D. Thesis, UCLA. Computer Science Department, 1978.

[2] Crocker, S., L. Marcus, and D. van-Mierop, *Microcode Verification Project Final Report*, USC/Information Sciences Institute WP-17, December 1979.

[3] Manna, Z., and R. Waldinger, "Is 'Sometime' sometimes better than 'Always'?" *Communications of the ACM*, 21(1978), 159-172.

[4] Pnueli, A., "The temporal logic of programs," *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pp. 46-57, October 1977.

[5] Pratt, V., "Semantical considerations on Floyd-Hoare logic," *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science*, pp. 109-121, October 1976.

[6] Saarinen, E., "Backwards-looking operators in tense logic and in philosophical analysis," in *Game-Theoretical Semantics*, E. Saarinen (ed.), pp. 215-244, D. Reidel, Dordrecht, 1978.